Junior Design Project Report

CSE 299

Outbreak Prediction Model by Machine Learning

Submitted By

1713013042 Umnoon Binta Ali

Supervisor

Tanjila Farah (Tnf)

Senior Lecturer and Lab Coordinator

Department of Electrical and Computer Engineering

North South University

ELECTRICAL AND COMPUTER ENGINEERING

NORTH SOUTH UNIVERSITY

[SPRING 2020]

Agreement Form

We take great pleasure in submitting our junior design project report on “Outbreak Prediction Model by Machine Learning”. This report is prepared as a requirement for junior design course. This course involves multidisciplinary teams of students who build and test custom designed systems, components or engineering processes. We would like to request you to accept this report as a partial fulfillment of Bachelor of Science degree under Electrical and Computer Engineering Department of North South University.

Declared By:

……………………………………………

Name: Umnoon Binta Ali

ID: 1713013042

Approved By:

……………………………

Supervisor

Tanjila Farah

Senior Lecturer and Lab Coordinator

Department of Electrical and Computer Engineering

North South University, Dhaka, Bangladesh

....

Dr. Rezaul Bari

Chair, Department of Electrical and Computer Engineering

North South University, Dhaka, Bangladesh

Outbreak Prediction Model by Machine Learning

In today's world, even with the development of technology, it's hard to keep track of natural epidemics. To ensure the global security, it's crucial to predict the future epidemics. For many years scientist have come together to come up with algorithm to get faster and more precise outcomes/results/accurate predictions of such outbreaks.

In last 10 years, we have seen the spread of Ebola, Dengue, Chikungunya, Influenza, Zika virus, Nipah virus, SARS, MARS and recently the deadly Novel Coronavirus (COVID-19) have taken lives all over the world. In the time of emergency, human beings have suffered for years to prevent such outbreaks. Things would not have gotten worse if there was some tool to predict when such outbreaks will hit the most.

With that aim in mind, we are proposing a model that analyzes the previous outbreak data and predicts the trend outbreaks of Covid-19 with precision.

Table of Contents

List of Figures:

CHAPTER 1
INTRODUCTION

1.1 Project Details:

Outbreak Prediction Model is such a platform where we take the raw data from a reliable source and manipulate the data in such a way that we can show visualization of different trend and forecast of its spread and establish a model to predict the estimated peak and estimated finish of that outbreak.

For limiting data resource we have only focused on recent outbreak Covid-19. The raw data has been collected from Johns Hopkins University's Github repository. They aggregated the data from sources like World Health Organization (WHO), European Centre for Disease Prevention and Control (ECDC) etc. So the data is reliable to make the model with precision.

The first feature that our model shows is the global visualization map where you can see the origin of Covid-19, which is Hubei states and the rest of the world as the virus spreads and the data grows in real time. In our model we can see global analysis and also analysis by country, date, states and month-year. Manipulating the data by date and breaking it down by countries we can see the virus spread for that territory on a particular date, month-year over a province and states. We have also been able to calculate the confirmed cases, number of deaths, number of recoveries, death rates, recovery rates for both globally and for a particular country/region. Then we calculated cumulative confirmed cases, cumulative deaths and also cumulative recovery cases. Then we further showed detailed analysis by breaking down the data by country in order to clearly see in which countries have maximum confirmed cases, death cases and recovery cases and vice versa. This way we can easy visualize in our Model for predicting the estimated peak and estimated finish of Covid-19.

The aim of this model is to give an estimation about when the coronavirus cases will be zero per Country and State. This model takes Hubei's trends and applies them in each state. Because Hubei's is the only state that has been successfully reached zero cases for a really long time. For our model, we assume that other countries and states will follow Hubei's trend. We assume that current conditions will remain the same. We do not take into account the following: Second wave, weather, population, social distancing nature, vaccine discovery.

1.2 Background and Motivation:

Background:

Due to big data progress and advancement of Machine learning in biomedical and healthcare communities, accurate study of medical data benefits early disease recognition, patient care and community services. [1] When the quality of medical data is incomplete the exactness of study is reduced. Moreover, different regions exhibit unique appearances of certain regional diseases, which may result in weakening the prediction of disease outbreaks. [2] In the proposed system, it provides machine learning algorithms for effective prediction of various disease occurrences in disease-frequent societies. It experiments the altered estimate models over real-life hospital data collected. [3] To overcome the difficulty of incomplete data, it uses a latent factor model to rebuild the missing data. It experiments on a regional chronic illness of cerebral infarction.

With the advance of big data analytics equipment, more devotion has been paid to disease expectation from the perception of big data inquiry, various explores have been conducted by choosing the features mechanically from a large number of data to improve the truth of menace classification rather than the formerly selected physiognomies. However, those prevailing work mostly measured structured data. Thus, risk organization based on big data analysis, the following tasks remain: How should the mislaid data be lectured? How can big data analysis expertise be used to estimate the disease?

To solve these problems, it sees the structured and unstructured data in healthcare field to assess the risk of disease. First, the system uses Naïve Bayes algorithm to generate the pattern and possible time of outbreak of disease. [1] This too fails to give us our required output. So we are going to use Linear Regression and Decision Tree Machine Learning Algorithm for our convenience.

Motivation:

To ensure the global security, it's crucial to predict the future epidemics. For many years scientist have come together to come up with algorithm to get faster and more precise outcomes/results/accurate predictions of such outbreaks. In last 10 years, we have seen the spread of Ebola, Dengue, Chikungunya, Influenza, Zika virus, Nipah virus, SARS, MARS and recently the deadly Novel Coronavirus (COVID-19) have taken lives all over the world. In the time of emergency, human beings have suffered for years to prevent such outbreaks. Things would not have gotten worse if there was some tool to predict when such outbreaks will hit the most. Today's world highly depend on data. The more data we can feed to our model the more accurate outcome we can get. Algorithms like Linear Regression and Decision Tree Algorithms help us to do just that. There's no room for inaccuracy. So our model will solve these problems

1.3 Project Goal:

Better Algorithm: Previously we have seen use of different types of algorithm approach used for specific data, problem and constraints for solve different types of problems. With regression and decision tree algorithm we get exactly the output we need.

Fast Outcomes: The model must provide the data within the shortest possible time. The chosen algorithm is proved to be the fastest algorithm for future prediction. It is faster to predict classes using this algorithm than many other classification algorithms.

Precision: As precision increases, the data points move closer to the regression line. Regression predictions are for the mean of the dependent variable. We need to quantify that scatter to know how close the predictions are to the observed values. This gives us results similar to real life.

User friendly platform: Our model can be accessible for both web and android users.

Prediction based on reliable sources: We will be collecting data for the dataset from reliable sources like – BBC, CNN, WHO, CDC, Google trends, Daily mail.

Real time aggregation: Sharing real-time aggregate statistics of private data has given much benefit to the public to perform Machine learning for understanding important phenomena, such as Influenza outbreaks and traffic congestion. [4] However, releasing time-series data with standard differential privacy mechanism has limited utility due to high correlation between data values.

CHAPTER 2

TECHNICAL DESIGN

2.1 Existing Solution:

Since Covid-19 is a completely new outbreak and there has been no machine learning model solely on this. But over the years a tons of papers have been published on predicting outbreaks. Focusing on those algorithms we will discuss why we chose linear regression and decision tree machine learning algorithms.

[1] Using structured and unstructured data from hospital it use Machine Learning Decision Tree algorithm and Map Reduce algorithm. To the best of their knowledge in the area of medical big data analytics none of the existing work focused on both data types. Compared to several typical estimate algorithms, the calculation exactness of their proposed algorithm reaches 94.8% with a convergence speed which is faster than that of the CNNbased unimodal disease risk prediction (CNN-UDRP) algorithm.

Disadvantage of this model is that it solely focuses on CNN reported data. So the data is limited and not fast enough. Even though this model uses decision tree, due to lack of data the outcome may not always come as we need it.

[3] Dengue outbreak is one of the top 10 diseases causing the most deaths worldwide. According to the World Health Organization, dengue infection has increased 30-fold globally over the past five decades. About 50–100 million new infections occur annually in more than 80 countries. Many researchers are working on measures to prevent and control the spread. One avenue of research is collaboration between computer science and the epidemiology researchers in developing methods of predicting potential outbreaks of dengue infection. An important research objective is to develop models that enable, or enhance, forecasting of outbreaks of dengue, giving medical professionals the opportunity to develop plans for handling the outbreak, well in advance. Researchers have been gathering and analyzing data to better identify the relational factors driving the spread of the disease, as well as the development of a variety of methods of predictive modeling using statistical and mathematical analysis and machine learning.

Advantages of this solution is that it analyzed: 1) the available data sources; 2) data preparation techniques; 3) data representations; 4) forecasting models and methods; 5) dengue forecasting models evaluation approaches; and 6) future challenges and possibilities in forecasting modeling of dengue outbreaks.

[5] In the recent studies on Covid-19, Liner Regression model has been used. To observe outbreak of COVID-19 in Henan province caused by the output population from Wuhan, and high-grade control measures were performed in Henan province, to study the phase of development and change of the epidemic in Henan province, and to make appropriate inferences about the influence of prevention and control measures and the phase of development of the epidemic. Linear regression analysis were used to

establish a linear regression model with the number of Wuhan roaming people as the dependent variable and the cumulative number of COVID-19 cases in Henan province as the dependent variable, and to calculate and plot the regional distribution of the number of cases in 18 cities in Henan province in accordance with the criteria of whether the number of cases exceeded the expected number.

This study only focuses on Wuhan confirmed cases and shows trends and forecasting f Wuhan region only. It doesn’t give the global forecasts or spread of the virus globally. But, it shows us how Linear Regression model can be used in our project and improvise it.

[6] COVID-2019 has been perceived as a worldwide pandemic and a few examinations are being led utilizing different numerical models to anticipate the likely advancement of this pestilence. These numerical models dependent on different factors and investigations are dependent upon potential inclination. Here, they presented a model that could be useful to predict the spread of COVID-2019. They have performed linear regression, Multilayer perceptron and Vector autoregression method for desire on the COVID-19 Kaggle data to anticipate the epidemiological example of the ailment and pace of COVID-2019 cases in India. Anticipated the potential patterns of COVID-19 effects in India dependent on data gathered from Kaggle. With the common data about confirmed, death and recovered cases across India for over the time length helps in anticipating and estimating the not so distant future.

Here, the dataset is not updated, so the inputs are limited. Hence, with the change is data, this model doesn’t change and give the outcomes in real time. This model also focuses on a fixed territory, India and not the global spread of the virus.

[7] Analysis of this disease requires major attention by the Government to take necessary steps in reducing the effect of this global pandemic. In this study, outbreak of this disease has been analysed for India till 30th March 2020 and predictions have been made for the number of cases for the next 2 weeks. SEIR model and Regression model have been used for predictions based on the data collected from John Hopkins University repository in the time period of 30th January 2020 to 30th March 2020. The performance of the models was evaluated using RMSLE and achieved 1.52 for SEIR model and 1.75 for the regression model. The RMSLE error rate between SEIR model and Regression model was found to be 2.01. Also, the value of R0 which is the spread of the disease was calculated to be 2.02. Expected cases may rise between 5000-6000 in the next two weeks of time. This study will help the Government and doctors in preparing their plans for the next two weeks.

2.2 Proposed Solution:

There are 3 types: Supervised, Unsupervised and Reinforcement

Supervised Learning:

This algorithm consist of a target / outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables). Using these set of variables, we generate a function that map inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression etc.

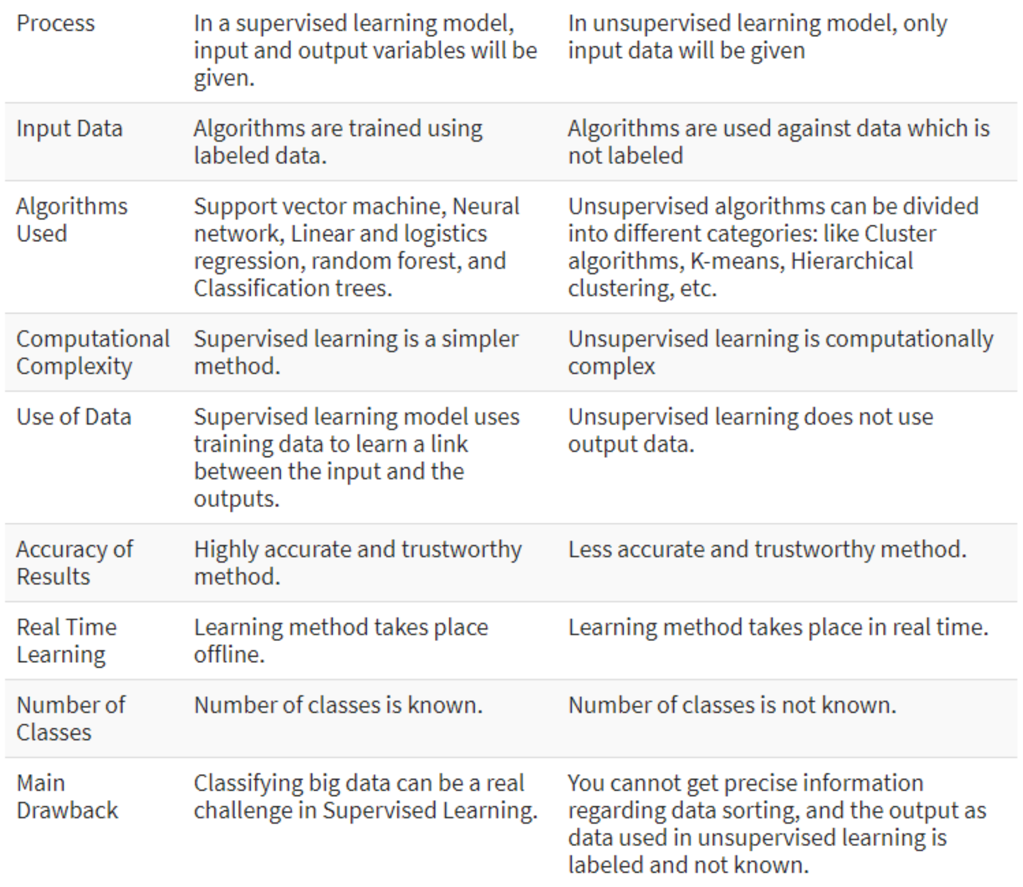
Unsupervised Learning:

In this algorithm, we do not have any target or outcome variable to predict / estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: Apriori algorithm, K-means.

Reinforcement Learning:

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Example of Reinforcement Learning: Markov Decision Process.

As our data are related to numbers and time-zone, we do not have any dependent variables. So we can only focus on the Supervised or Unsupervised Algorithms. But we chose supervised algorithm because of the following reasons.



Process	In a supervised learning model, input and output variables will be given.	In unsupervised learning model, only input data will be given
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data which is not labeled
Algorithms Used	Support vector machine, Neural network, Linear and logistics regression, random forest, and Classification trees.	Unsupervised algorithms can be divided into different categories: like Cluster algorithms, K-means, Hierarchical clustering, etc.
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Use of Data	Supervised learning model uses training data to learn a link between the input and the outputs.	Unsupervised learning does not use output data.
Accuracy of Results	Highly accurate and trustworthy method.	Less accurate and trustworthy method.
Real Time Learning	Learning method takes place offline.	Learning method takes place in real time.
Number of Classes	Number of classes is known.	Number of classes is not known.
Main Drawback	Classifying big data can be a real challenge in Supervised Learning.	You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known.

Fig-1. Comparison between supervised and unsupervised Learning

Regression:

Regression technique predicts a single output value using training data.

Example: You can use regression to predict the house price from training data. The input variables will be locality, size of a house, etc.

Classification:

Classification means to group the output inside a class. If the algorithm tries to label input into two distinct classes, it is called binary classification. Selecting between more than two classes is referred to as multiclass classification.

Example: Determining whether or not someone will be a defaulter of the loan.

Strengths:

Outputs always have a probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.

Weaknesses:

Logistic regression may underperform when there are multiple or non-linear decision boundaries. This method is not flexible, so it does not capture more complex relationships.

Decision Tree Algorithm:

Decision Tree is a very popular machine learning algorithm. Decision Tree solves the problem of machine learning by transforming the data into tree representation. Each internal node of the tree representation denotes an attribute and each leaf node denotes a class label. Decision tree algorithm can be used to solve both regression and classification problems.

Strengths:

Compared to other algorithms decision trees requires less effort for data preparation during pre-processing. A decision tree does not require normalization of data. A decision tree does not require scaling of data as well. Missing values in the data also does not affect the process of building decision tree to any considerable extent. A Decision trees model is very intuitive and easy to explain to technical teams as well as stakeholders.

Weaknesses:

A small change in the data can cause a large change in the structure of the decision tree causing instability. For a Decision tree sometimes calculation can go far more complex compared to other algorithms. Decision tree often involves higher time to train the model. Decision tree training is relatively expensive as complexity and time taken is more. Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

2.3 Contribution

In section 2.1, we have seen the use of Linear Regression model in terms of visualizing the spread of virus, Covid-19 and different analysis including trend and forecasting of confirmed cases, death cases and recovered cases. We have also seen the cumulative calculation of the confirmed cases, deaths and recovery cases. But those studies have solely focused on a definite territory like Wuhan and India. They do not represent the global dataset.

Using the same strategy we can get the global analysis of the spread of virus, the cumulative analysis and also the analysis by country and states. Then we can predict the peak of confirmed cases by country and estimated finish of the virus by country. To do so, we improvised the existing model, Linear Regression with Decision Tree algorithm.

2.4 Technical Design: Module Level

In this section we will take each feature and see how the code works.

#uploading Epidemio banner

- import os
- from IPython.display import Image
- PATH = "C:\\Users\\HP\\"
- Image(filename = PATH + "banner.jpg", width=900, height=900)

#Loading pakages and libraries

- import os #provides functions for interacting with the operating system
- import numpy as np
- import pandas as pd
- from datetime import datetime

#Uploading the datasets

- data_confirmed = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_confirmed_global.csv')
- data_deaths = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_deaths_global.csv')
- data_Recovered = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_recovered_global.csv')

- print("The Shape of Cornirmed is: ", data_confirmed.shape)
- print("The Shape of Death is: ", data_deaths.shape)
- print("The Shape of Recovered is: ", data_Recovered.shape)

#unpivoting the date

- data_confirmed2 = pd.melt(data_confirmed, id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'], var_name=['Date'])
- data_deaths2 = pd.melt(data_deaths, id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'], var_name=['Date'])
- data_Recovered2 = pd.melt(data_Recovered, id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'], var_name=['Date'])

- print("The Shape of Cornirmed is: ", data_confirmed2.shape)
- print("The Shape of deaths is: ", data_deaths2.shape)
- print("The Shape of recovered is: ", data_Recovered2.shape)

# Converting the new column to dates

- data_confirmed2['Date'] = pd.to_datetime(data_confirmed2['Date'])
- data_deaths2['Date'] = pd.to_datetime(data_deaths2['Date'])
- data_Recovered2['Date'] = pd.to_datetime(data_Recovered2['Date'])

# Renaming the Values

- data_confirmed2.columns = data_confirmed2.columns.str.replace('value', 'Confirmed')
- data_deaths2.columns = data_deaths2.columns.str.replace('value', 'Deaths')
- data_Recovered2.columns = data_Recovered2.columns.str.replace('value', 'Recovered')

# Investigating the NULL values

- data_Recovered2.isnull().sum()

# Dealing with NULL values

- data_confirmed2['Province/State'].fillna(data_confirmed2['Country/Region'], inplace=True)
- data_deaths2['Province/State'].fillna(data_deaths2['Country/Region'], inplace=True)
- data_Recovered2['Province/State'].fillna(data_Recovered2['Country/Region'], inplace=True)

# printing shapes before the join

- print("The Shape of Cornirmed is: ", data_confirmed2.shape)
- print("The Shape of Cornirmed is: ", data_deaths2.shape)
- print("The Shape of Cornirmed is: ", data_Recovered2.shape)

# Full Joins
# Confirmed with Deaths

-
  ```
  full_join =
  data_confirmed2.merge(data_deaths2[['Province/State','Country/Region','Date','Deaths']],
                      how = 'left',
                      left_on = ['Province/State','Country/Region','Date'],
                      right_on = ['Province/State', 'Country/Region','Date'])
  ```

- print("Shape of first join: ", full_join.shape)

# full join with Recovered

- full_join = full_join.merge(data_Recovered2[['Province/State','Country/Region','Date','Recovered']],

  how = 'left',

  left_on = ['Province/State','Country/Region','Date'],

  right_on = ['Province/State', 'Country/Region','Date'])

- print("Shape of second join: ", full_join.shape)

# checking for null values

- full_join.isnull().sum()

# Adding Month and Year as a new Column

- full_join['Month-Year'] = full_join['Date'].dt.strftime('%b-%Y')

#breaking the dataset by day
#filtering data for Anhui

- test = full_join[full_join['Province/State'] == 'Anhui']
- full_join2 = test.copy()

#creating a new date columns - 1

- full_join2['Date - 1'] = full_join2['Date'] + pd.Timedelta(days=1)
- full_join2.rename(columns={'Confirmed': 'Confirmed - 1', 'Deaths': 'Deaths - 1', 'Recovered': 'Recovered - 1',

  'Date': 'Date Minus 1'}, inplace=True)

- full_join3 = test.merge(full_join2[['Province/State', 'Country/Region','Confirmed - 1', 'Deaths - 1',

  'Recovered - 1', 'Date - 1', 'Date Minus 1']], how = 'outer',

  left_on = ['Province/State','Country/Region','Date'],

```
      right_on = ['Province/State', 'Country/Region','Date - 1'])

•   full_join3['Confirmed Daily'] = full_join3['Confirmed'] -
    full_join3['Confirmed - 1']
```

```
#Filtering all the dataset
```

```
•   full_join2 = full_join.copy()

•   full_join2['Date - 1'] = full_join2['Date'] + pd.Timedelta(days=1)
•   full_join2.rename(columns={'Confirmed': 'Confirmed - 1', 'Deaths': 'Deaths -
    1', 'Recovered': 'Recovered - 1',
                'Date': 'Date Minus 1'}, inplace=True)


•   full_join3 = full_join.merge(full_join2[['Province/State',
    'Country/Region','Confirmed - 1', 'Deaths - 1',
                 'Recovered - 1', 'Date - 1', 'Date Minus 1']], how = 'left',
                  left_on = ['Province/State','Country/Region','Date'],
                  right_on = ['Province/State', 'Country/Region','Date - 1'])

•   full_join3['Confirmed Daily'] = full_join3['Confirmed'] -
    full_join3['Confirmed - 1']
•   full_join3['Deaths Daily'] = full_join3['Deaths'] - full_join3['Deaths - 1']
•   full_join3['Recovered Daily'] = full_join3['Recovered'] - full_join3['Recovered
    - 1']
```

```
# Additing manually the numbers for first day
```

```
•   full_join3['Confirmed Daily'].loc[full_join3['Date'] == '2020-01-22'] =
    full_join3['Confirmed']
•   full_join3['Deaths Daily'].loc[full_join3['Date'] == '2020-01-22'] =
    full_join3['Deaths']
•   full_join3['Recovered Daily'].loc[full_join3['Date'] == '2020-01-22'] =
    full_join3['Recovered']
```

```
# deleting columns
```

```
•   del full_join3['Confirmed - 1']
•   del full_join3['Deaths - 1']
•   del full_join3['Recovered - 1']
•   del full_join3['Date - 1']
•   del full_join3['Date Minus 1']
```

Exporting the data

- path = "C:\\Users\\HP\\"
- os.chdir(path)

- full_join3.to_csv('PowerBI', sep='\t')

#Loading pakages and libraries

- import os #provides functions for interacting with the operating system
- import numpy as np
- import pandas as pd
- from matplotlib import pyplot as plt
- import seaborn as sns
- from sklearn.linear_model import LinearRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import r2_score, explained_variance_score, mean_absolute_error, mean_squared_error
- from math import sqrt
- from datetime import datetime
- import random

- %matplotlib inline

#to change scientific numbers to floar

- np.set_printoptions(formatter={'float_kind':'{:f}'.format})

#incresing the size of sns plot

- sns.set(rc={'figure.figsize':(12,10)})

- pd.set_option('display.max_rows',500)
- pd.set_option('display.max_columns',500)
- pd.set_option('display.width',1000)

#Loading the cumulative raw data

- data_confirmed = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_confirmed_global.csv')
- data_deaths = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_deaths_global.csv')
- data_Recovered = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_recovered_global.csv')
- data_confirmed_us = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_confirmed_US.csv')
- data_deaths_us = pd.read_csv('C:\\Users\\HP\\COVID-19-master\\csse_covid_19_data\\csse_covid_19_time_series\\time_series_covid19_deaths_US.csv')
- print("The Shape of Cornirmed is: ", data_confirmed.shape)
- print("The Shape of deaths is: ", data_deaths.shape)
- print("The Shape of recovered is: ", data_Recovered.shape)
- print("The Shape of Cornirmed US is: ", data_confirmed_us.shape)
- print("The Shape of deaths Us is: ", data_deaths_us.shape)

#creating dates list
#dropping columns we do not need
#Averaging the Lat & Long_
#Summing the values
#left joining the Average Lat and Long
#Rordering the columns

- column_dates=data_confirmed_us.columns

column_dates=column_dates.drop(labels=['UID','iso2','iso3','code3','FIPS','Admin2','Combined_Key','Province_State','Country_Region','Lat','Long_'])

- data_confirmed_us.drop(['UID','iso2','iso3','code3','FIPS','Admin2','Combined_Key'], axis=1, inplace=True)
- data_deaths_us.drop(['UID','iso2','iso3','code3','FIPS','Admin2','Combined_Key'], axis=1, inplace=True)

- Lat_lon_Avg=data_confirmed_us.groupby(['Province_State','Country_Region'],as_index=False).agg({'Lat':'median','Long_':'median'})

- data_confirmed_us= data_confirmed_us.groupby(['Province_State','Country_Region'], as_index= False)[column_dates].sum()
- data_deaths_us= data_deaths_us.groupby(['Province_State','Country_Region'], as_index= False)[column_dates].sum()

- data_confirmed_us= data_confirmed_us.merge(Lat_lon_Avg, how = 'left', left_on = ['Province_State','Country_Region'], right_on = ['Province_State','Country_Region'])
- data_deaths_us = data_deaths_us.merge(Lat_lon_Avg, how = 'left', left_on = ['Province_State','Country_Region'], right_on = ['Province_State','Country_Region'])

- data_confirmed_us= data_confirmed_us.rename(columns={'Province_State':'Province/State','Country_Region':'Country/Region','Long_':'Long'})
- data_deaths_us= data_confirmed_us.rename(columns={'Province_State':'Province/State','Country_Region':'Country/Region','Long_':'Long'})

- data_confirmed_us=data_confirmed_us[data_confirmed_us.columns]
- data_deaths_us=data_deaths_us[data_confirmed_us.columns]

#loading calender

- calendar = pd.read_csv('C:\\Users\\HP\\Calendar.csv')

- calendar['Date']= pd.to_datetime(calendar['Date'], dayfirst=True)
- print(full_join3.shape)

- full_join4 = full_join3.merge(calendar, how = 'left', left_on = 'Date', right_on = 'Date')
- print(full_join4.shape)

#investigating Hubei Alone

- hubei = full_join4[full_join4['Province/State']=='Hubei']
- hubei.head()

- #hubei peak confirmed cases

peak = hubei['Confirmed Daily'].max()

- #hubei peak date_id

peak_date_id = hubei['Date_id'][hubei['Confirmed Daily']==hubei['Confirmed Daily'].max()]

- #hubei dataset after peak to find when it hits zero

dataset_after_peak = hubei[hubei['Date_id']>=peak_date_id.max()]

- #removing zeros to select max datapoint

ending_day=dataset_after_peak[dataset_after_peak['Confirmed Daily']>1]

- #selecting max date_id

ending_day = ending_day[ending_day['Date_id']==ending_day['Date_id'].max()]

- #offsetting the month_id by 1 and saving it. Because, it came to zero a day after

ending_day_date=ending_day['Date'] + pd.DateOffset(1)

- #resetting the ending dataset with offset

ending_day = dataset_after_peak[dataset_after_peak['Date']==ending_day_date.max()]

- #saving the final date_id

ending_date_id = ending_day['Date_id']

- #visualizing

ending_day

#calculating how many days it took Hubei to go from current date to zero and how many days it took Hubei to actually hit the current peak

- days_until_conf_goes_zero = ending_date_id.max() - peak_date_id.max()
- days_until_hit_peak = peak_date_id.max()-1

- days_until_conf_goes_zero
- print("It took Hubei ", days_until_hit_peak, "days to reach peak")
- print("It took Hubei",days_until_conf_goes_zero, "days to go back to Zero")

#Prediction Model

```
states = full_join4['Province/State'][full_join4['Confirmed']>20].unique()
states=['Cyprus']

try:
   del final_df
```

```
except:
    print("")

for s in states:
    dataset = full_join4[full_join4['Province/State'] == s]
    print(s)

    #peak number
    peak = dataset['Confirmed Daily'].max()

    #peak date id
    peak_date_id = dataset['Date_id'][dataset['Confirmed Daily']== peak]
    peak_date_id = peak_date_id.max()

    #peak date
    peak_date = dataset['Date'][dataset['Date_id']== peak_date_id.max()]
    peak_date=peak_date.max()

    #after peak dataset
    dataset_after_peak = dataset[dataset['Date_id']>=peak_date_id.max()]

    #sorting the df by date
    dataset = dataset.sort_values(by=['Date'],ascending=True)

    #filtering the dataset before the peak
    dataset_before_peak = dataset[dataset['Date_id']< peak_date_id.max()]

    #filtering the dataset to exclude any cumulative confirm case less than 20 which
will be the starting point of the graph
    dataset_curve_starting_point =
dataset_before_peak[dataset_before_peak['Confirmed']<20]
    dataset_curve_starting_point= dataset_curve_starting_point.tail(1)

    try:
        dataset_curve_starting_point_date = dataset_curve_starting_point['Date'].iloc[0]
        dataset_curve_starting_point_date_id =
dataset_curve_starting_point['Date_id'].iloc[0]
    except:
        dataset_curve_starting_point_date=dataset_before_peak['Date'].min()
        dataset_curve_starting_point_date_id=dataset_before_peak['Date_id'].min()

    #calculating the estimated peak based on Hubei
    est_peak_date_id= dataset_curve_starting_point_date_id + days_until_hit_peak
    est_peak_date=calendar['Date'][calendar['Date_id']==est_peak_date_id]

    #calculating the days until current peak
    days_until_current_peak = peak_date_id - dataset_curve_starting_point_date_id

    #calculating the est finish based on current peak
    Est_finish_current_peak_date_id = peak_date_id + days_until_conf_goes_zero

Est_finish_current_peak_date=calendar['Date'][calendar['Date_id']==Est_finish_curre
nt_peak_date_id.max()]

    #calculating the Est finish based on est peak
    Est_finish_based_on_est_peak_date_id = est_peak_date_id +
days_until_conf_goes_zero
    Est_finish_based_on_est_peak_date=calendar['Date'][calendar['Date_id']==
Est_finish_based_on_est_peak_date_id.max()

    #output by country
    peak_data_by_state = pd.DataFrame({'Province/State': [s], 'Starting Date (c>30)':
[dataset_curve_starting_point_date],
                              'Starting Date ID
(c>30)':dataset_curve_starting_point_date_id, 'Peak Date':peak_date,
                              'Peak Date ID':peak_date_id, 'Days until peak for
Hubei':days_until_hit_peak,
                               'Days Until Peak for this
State/Country':days_until_current_peak,
                               'Est Date for Peak':est_peak_date.iloc[0], 'Est Date ID for
Peak':est_peak_date_id,
                              'Est Finish Date Based on current
Peak':Est_finish_current_peak_date.iloc[0],
                              'Est Finish Date ID Based on current
Peak':Est_finish_current_peak_date_id,
                              'Est Finish Date Based on Est
peak':Est_finish_based_on_est_peak_date.iloc[0],
                              'Est Finish Date ID Based on Est
peak':Est_finish_based_on_est_peak_date_id}, index=[0])

    try:
        final_df = pd.concat([final_df, peak_data_by_state], ignore_index = True)
    except:
        final_df = peak_data_by_state
```

#Error check in dataframe

- final_df.groupby('Province/States')['Province/State'].count()

#creating a cross join function

- def cross_join(table1,table2):
```
return (table1.assign(key=1).merge(table2.assign(key=1), on='key').drop('key',1))
```

#Estimation

```
states = full_join4['Province/State'][full_join4['Confirmed']>20].unique()
states=['Cyprus']

try:
   del final_cross_join
except:
   print("")

for s in states:
   print(s)
   dataset2=full_join4[full_join4['Province/State'] == s]
   dataset_with_totals=final_df[final_df['Province/State']==s]

   #14 day Average
   conf_last_14_days = dataset2['Confirmed Daily'].tai(14).mean()
   conf_last_14_days

   #current date
   current_date= dataset2['Date'].max()
   current_date_id=calendar['Date_id'][calendar['Date']==current_date]

   #filtering the calendar data we need
   calendar_data_after_today =
calendar[['Date','Date_id']].loc[calendar['Date_id']>current_date_id.max()]

   #creating table s for cross joining
   table_a=dataset2[['Province/State','Country/Region']].drop_duplicates()

   #cross join
   cross_join_1 = cross_join(table_a, calendar_data_after_today)

   #estimate confirmed cases until estimate peak
   cross_join_1['Est confirm cases until Est peak'] = 0
   cross_join_1['Est confirm cases until Est peak'][cross_join_1['Date_id'] <=
dataset_with_totals['Est Date ID for Peak'].max()] = conf_last_14days
```

```
    #days between Est peak to Est finish
    days_between_est_peat_to_est_finish = datset_with_totals['Est Finish Date ID
Based on Est Peak'].max() - dataset_with_totals['Est Date ID for Peak'].max()

    #calculate the actual values
    conf_cases=np.linspace(conf_last_14days,0,(days_between_est_peat_to_est_finish-
1))
    a= dataset_with_totals['Est Date ID for Peak'].max()+1

    #creating a DF
    df1= pd.DataFrame({'Date_id': range(a,dataset_with_totals['Est Finish Date ID
Based on Est Peak'].max()), 'Est confirm cases from Est Peak to Est
End':conf_cases})

    #joing the DF with the cross join

    cross_join_1 = cross_join_1.merge(df1,how='left', on ='Date_id')

    #selecting the data from current peak ans onwards
    current_peak_value_start = dataset2['Confirmed
Daily'][dataset2[Date_id]==dataset_with_totals['Peak Date ID'].max().astype(int)]
    current_peak_value_start = current_peak_value_start.max() #new

    #Creating range
    example_range = range(dataset_with_totals['Peak Date ID'].max().astype(int),
dataset_with_totals['Est Finish Date ID Based on current Peak'].max().astype(int))

    #difference in days
    value_bins = dataset_with_totals['Est Finish Date ID Based on current
Peak'].max().astype(int) - dataset_with_totals['Peak Date ID'].max().astype(int)

    #calculate the Est Confirm Cases from current Peak to Est End
    values = np.linspace(current_peak_value_start,0,(value_bins))
    values = values.ravel()

    #creating a df
    try:
        df2= pd.DataFrame({'Date_id':example_range, 'Est Confirm Cases from current
Peak to Est End':values})
    except:
        df2= pd.DataFrame({'Date_id':[0], 'Est Confirm Cases from current Peak to Est
End':[0]})

    #joining the df2 with cross join
    cross_join_1 = cross_join_1.merge(df2,how='left', on='Date_id')
     try:
```

```
    final_cross_join = pd.concat([final_cross_join, cross_join_1])
except:
    final_cross_join = cross_join_1
```

```
#Exporting dataset
```

- path = "C:\\Users\\HP\\"
- os.chdir(path)

- full_join3.to_csv('PowerBI', sep='\t')

2.5 Technical Design: System Level

Now the dataset is ready to be uploaded for visualization. So, we first import the dataset in our Powerbi tool then we make all the graphs and charts accordingly.
First we make the filter bar containing the following

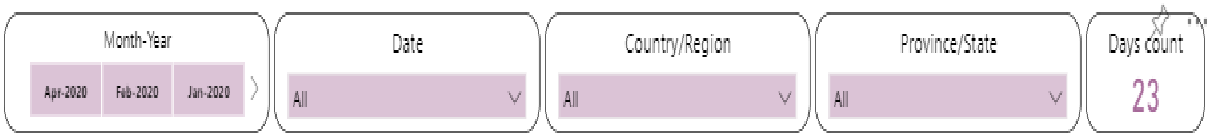


Fig-2: Filter Bar of Visualization.

Then we load the map for global and filtered visualizations. The red flag represents the

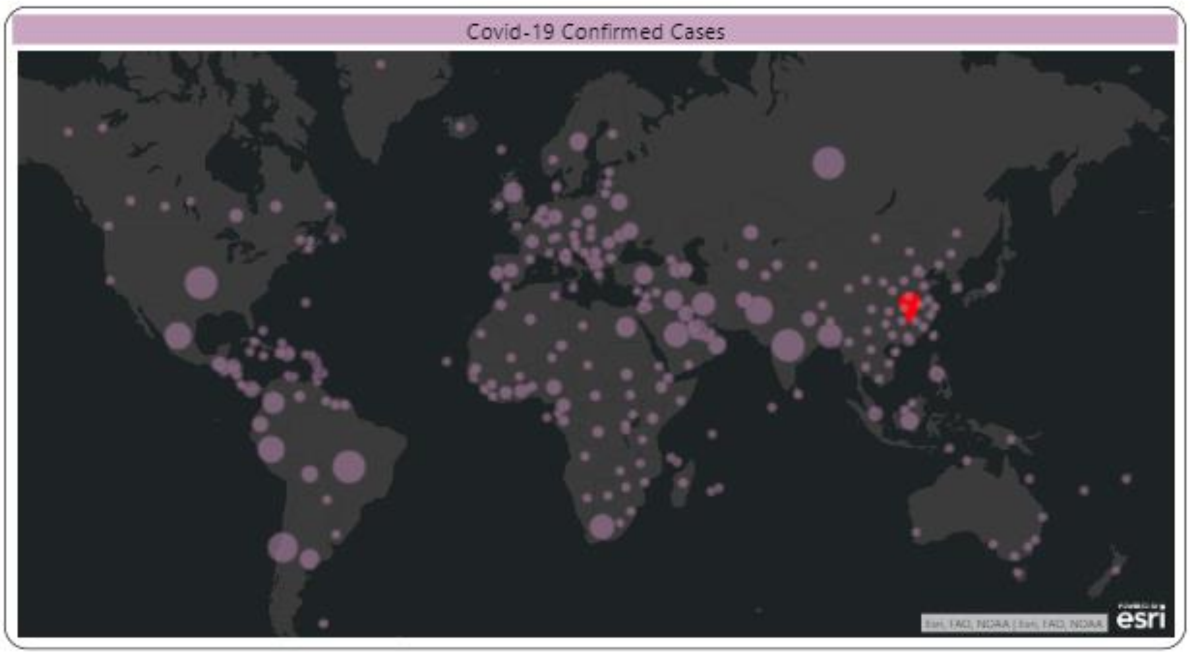


Fig-3: Global Visualization of Confirmed Cases and Origin of the Outbreak origin of the virus, Covid-19, which is Wuhan. And the circles represent the confirmed cases. The larger the circle, the larger is the number of confirmed cases. The smaller the circle the smaller the confirmed cases at that country or state. This map is updated every 24 hours as the Johns Hopkins University once a day in real time.

Then summary panel is made containing total number of confirmed cases, deaths and recoveries. The death rate is calculated by dividing the total confirmed cases with the confirmed cases daily and expressed as percentage with two significant figure. Similarly the death rate is calculated by dividing the total deaths with the deaths daily and expressed as percentage with two significant figures.

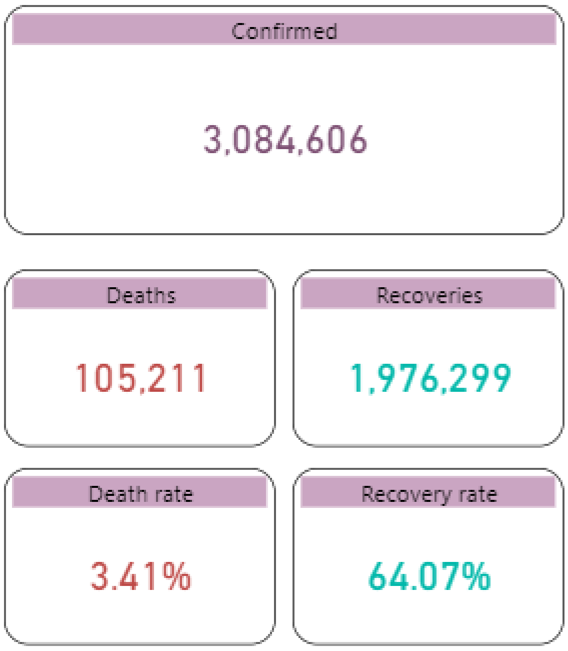


Fig-4: Summary Dashboard.

In the following diagram, we see the global analysis of Covid-19. In the first three Line graph, we see that the confirmed cases have increased rapidly from the end of March and to April 2020. The recovery cases have ups and downs. But it has also increased over the period of the month April. On the other hand the number of death cases have been increased a lot in April. The shadows in the upper and lower side of the graph represents how up and how low the numbers can possibly go. It’s basically the upper bound and the lower bound of the actual number.

The three graphs below, shows the cumulative calculation of the confirmed cases, the recovery cases and the death cases. These are all global values and not by country. From here we see that, the cumulative number increases as we go from March to April. It is because, the global confirmed cases, death cases and recovery cases increased in the month of April worldwide.

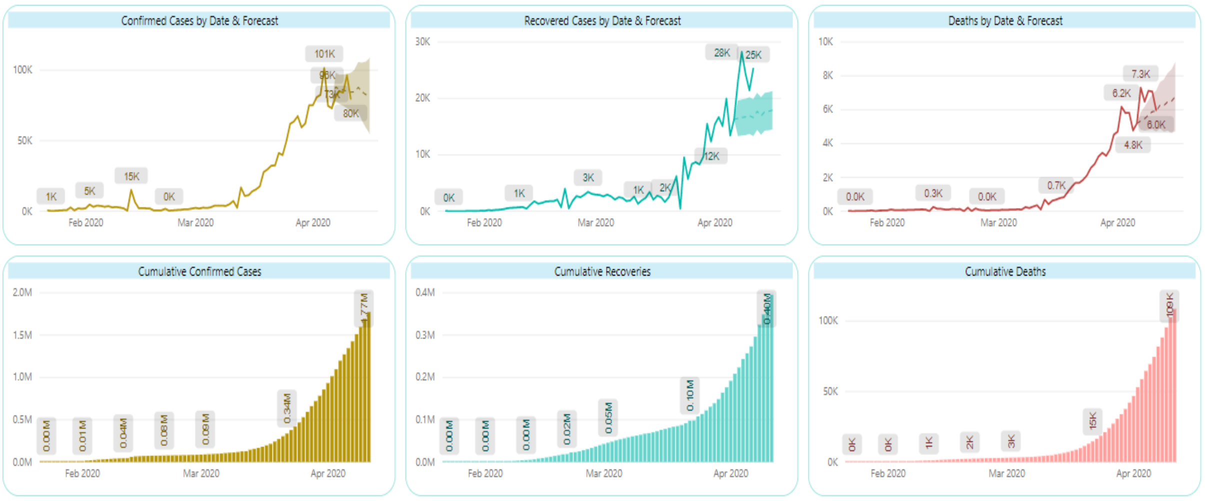
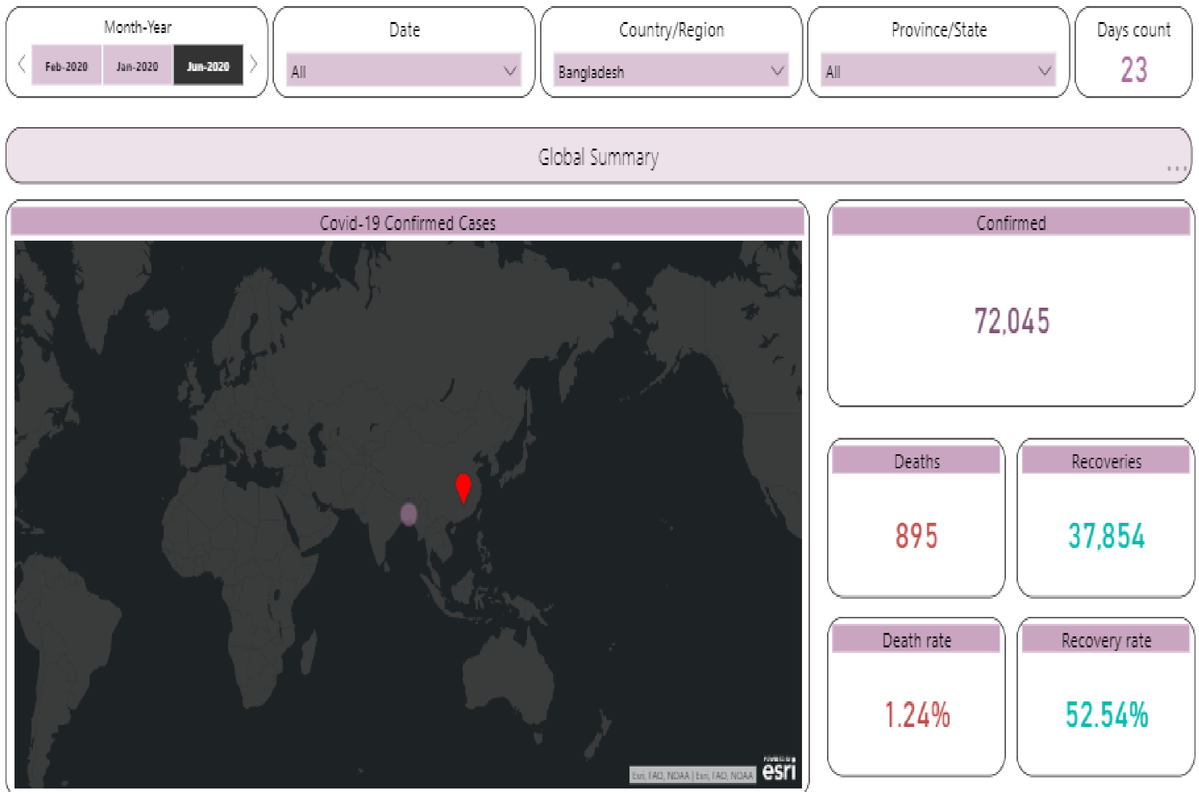


Fig-5: Global Cumulative Analysis

From here we can filter the data as we want. For example, the following data is filtered out for Bangladesh in the month of June.



Month-Year: Feb-2020 | Jan-2020 | Jun-2020

Date: All

Country/Region: Bangladesh

Province/State: All

Days count: 23

Global Summary

Covid-19 Confirmed Cases

Confirmed: 72,045

Deaths: 895

Recoveries: 37,854

Death rate: 1.24%

Recovery rate: 52.54%

Fig-6: Data Summary Dashboard for Bangladesh.

Here, we have selected June-2020 to visualize the data for Bangladesh only. So the Dashboard is refreshed automatically. We can see the circle on Bangladesh and the analysis beside it. The total confirmed cases is 72,045, this is the cumulative confirmed number from the start till June of 2020. As the number increases, the circle on the map widens in real time. We can also see that total number of deaths are 895 and total number of recoveries are 37,854 till the beginning of June-2020. These are all cumulative values.

The following diagram shows the prediction of the highest peak, estimated peak and estimated finish of the outbreak for Bangladesh which was taken on the month of April.

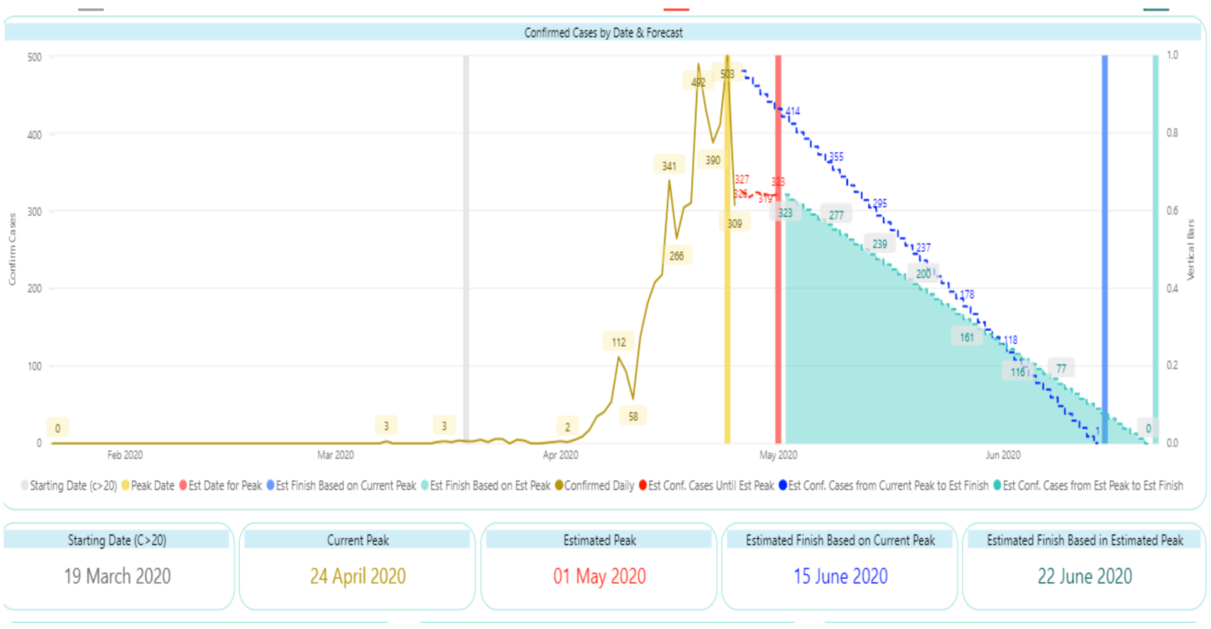


Fig-7: Prediction Model Visualization

We have created a custom model that models the trends of Wuhan (virus birth) in order to apply them on other states and countries. The reason we used Wuhan is because it's the only country that managed to bring their cases down to zero. Assuming we trust their numbers. The aim of this model is to predict when (date) coronavirus confirm cases will hit zero by state and country and what will the numbers look like. The aim of this model is to give an estimation about when the coronavirus cases will be zero per Country and State. This model takes Hubei's trends and applies them in each state. Because Hubei's is the only state that has been successfully reached zero cases for a really long time. For our model, we assume that other countries and states will follow Hubei's trend. We assume that current conditions will remain the same. We do not take into account the following: Second wave, weather, population, social distancing nature, vaccine discovery.

The grey colored vertical line represents the time when the first 20 confirmed case was found. The yellow colored vertical line represents the current peak of the virus. The red colored vertical line represents the estimated current peak of the virus. This is our prediction that, considering the Hubei trend and the values of that country, we estimate to hit the peak of confirmed cases at the red line. The dark blue line represents our prediction of estimated finish based on the current peak. So, if we assume that, the yellow line peak is the current peak and it will not have another peak. Then, the estimated finish of the outbreak will be at the dark blue line. Here, the figure was taken at the end of April where the peak was and considering there will be no other peak the estimated finish down to zero of the outbreak was the end of the June. The last light blue line is the estimated finish based on estimated peak. Given that the country followed Hubei’s trend and the estimated peak was at the red line then we can predict that the estimated finish will be at the light blue vertical line.

The following figures shows the trends and forecasting of the data by country, Bangladesh.

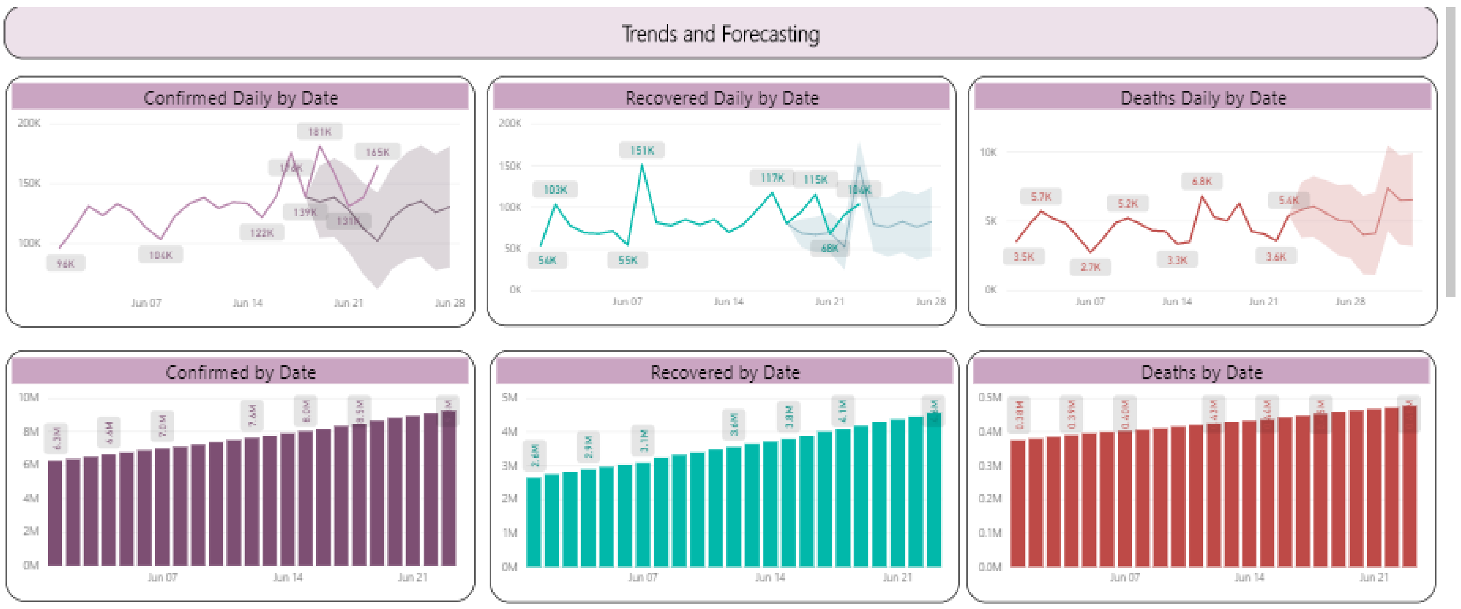
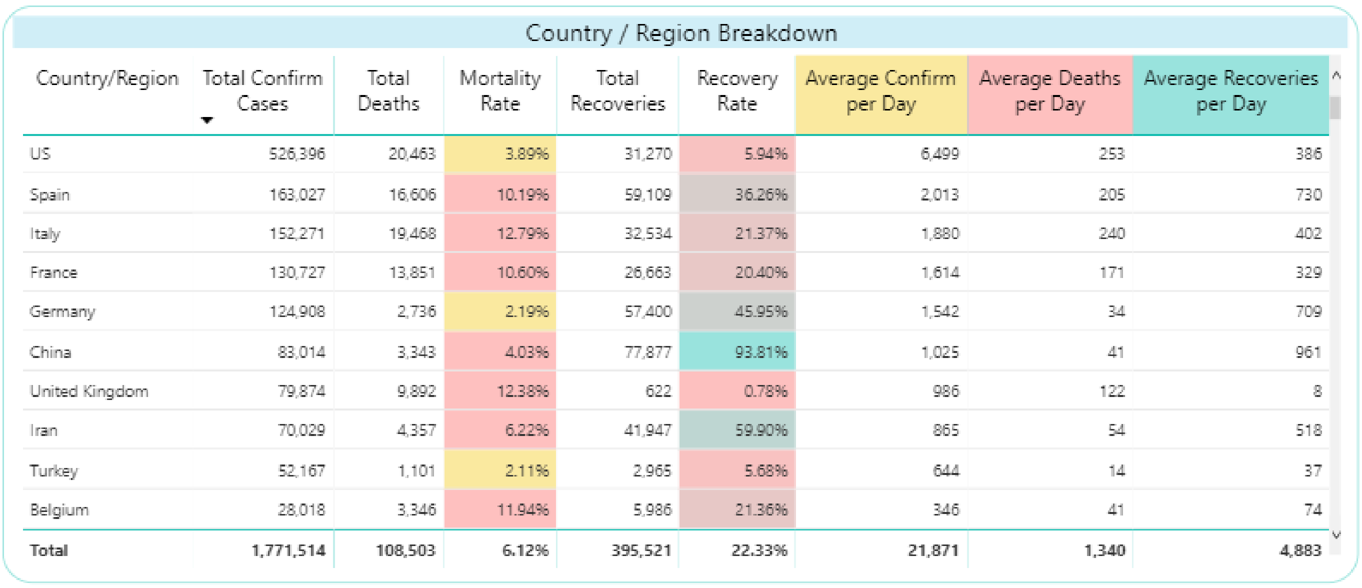


Fig-8: Cumulative Analysis for Bangladesh.



Country / Region Breakdown

Country/Region	Total Confirm Cases	Total Deaths	Mortality Rate	Total Recoveries	Recovery Rate	Average Confirm per Day	Average Deaths per Day	Average Recoveries per Day
US	526,396	20,463	3.89%	31,270	5.94%	6,499	253	386
Spain	163,027	16,606	10.19%	59,109	36.26%	2,013	205	730
Italy	152,271	19,468	12.79%	32,534	21.37%	1,880	240	402
France	130,727	13,851	10.60%	26,663	20.40%	1,614	171	329
Germany	124,908	2,736	2.19%	57,400	45.95%	1,542	34	709
China	83,014	3,343	4.03%	77,877	93.81%	1,025	41	961
United Kingdom	79,874	9,892	12.38%	622	0.78%	986	122	8
Iran	70,029	4,357	6.22%	41,947	59.90%	865	54	518
Turkey	52,167	1,101	2.11%	2,965	5.68%	644	14	37
Belgium	28,018	3,346	11.94%	5,986	21.36%	346	41	74
Total	**1,771,514**	**108,503**	**6.12%**	**395,521**	**22.33%**	**21,871**	**1,340**	**4,883**

Fig-9: Breakdown of Data by Country.

In the above figure, we see the breakdown of data by country. At the left hand side of it, there is a list of countries and with the total number of confirmed cases of each country. This table is also updated in real time. Then we see the total number of deaths, mortality rate, total recoveries and recovery rate by country. Then we find the average of the confirmed cases, deaths and recovery cases per day by country. This helps us to form the following table which shows the global average count of confirmed cases, recovery cases and death cases per day. Then we proceed to see which country has the maximum/minimum confirmed cases.

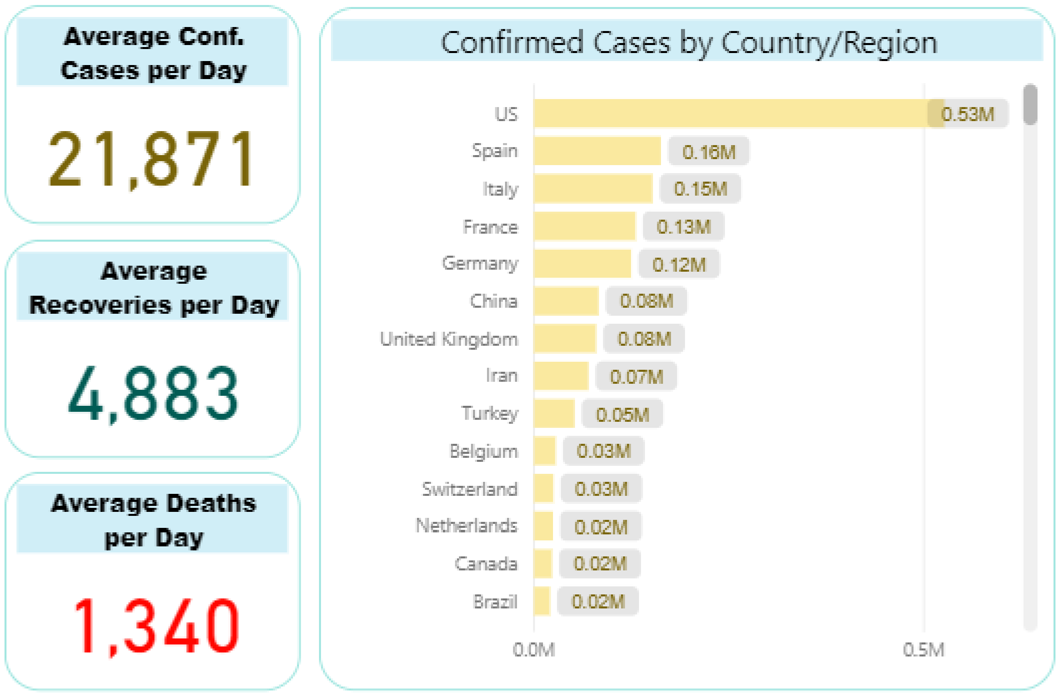


Fig-10: Global Average Analysis.

2.6 Required Skill

To build this model we need to learn Python and how to manipulate data into Jupyter Notebook anaconda. Also we need to learn the basics for PowerBi, as this is our visualization tool for the project.

CHAPTER 3

Working Timeline

Date	Task
26/01/20	Introduction
02/02/20	Topic selection
09/02/20	Summarize 5 previous research paper related to the selected topic
16/02/20	Project proposal Submission
23/02/20	Learning Naïve Bayes algorithm and start making dataset
01/03/20	Creating Webapp and Android app interface
08/03/20	Complete dataset by Web scraping and including it to the database
15/03/20	First trial of the model
22/03/20	Second trial of the model
29/03/20	Finalizing the model
05/04/20	Poster and final documentation
12/04/20	Publishing the system and project presentation

CHAPTER 4

Project Summary

4.1 Result and Discussion:

Outbreak Prediction Model is a common platform for predicting future outbreaks of multiple diseases as accurately as possible. Over the years we have observed that for lack of conscious and precautionary measures many countries have suffered greatly and thousands of people have lost their lives. But there is no convenient system which involve the people directly so that they too can do the necessary things in time of emergency. This model offers just that.

Our model shows statistics and real time aggregation of data which are collected from reliable sources from around the globe. Some of the new features that we are going to implement for our predicting system are- Users will be able to make their own profile. Users will be able to take part in daily survey. forecasting virus outbreak. Showing precise location of the outbreak on map. Showing 10 past 5 years history of virus outbreaks. Showing statistics and graph related to the information's of various outbreaks. Notify on major outbreaks. Notify about the preventions and precautions.

4.2 Problem faced and solution

There are many countries that did not release the number of recoveries, confirmed cases and death cases. For that reason, the prediction model may not always give the outcome as accurate as in real life. But the time of hitting the peak and finishing time remains the same as the prediction given that all the countries followed Hubei's trend.

4.3 Future Development

There's a lot of room for development. The algorithm can always be developed for better prediction with more data. The more data we feed the machine the more precise the output will be.

We can also put different datasets in our model and by manipulating them in the same way we can predict the future outbreak of diseases.

4.5 Conclusion

In today’s world, even with the development of technology, it’s hard to keep track of natural epidemics. To ensure the global security, it’s crucial to predict the future epidemics. For many years scientist have come together to come up with algorithm to get faster and more precise outcomes/results/accurate predictions of such outbreaks. In last 10 years, we have seen the spread of Ebola, Dengue, Chikungunya, Influenza, Zika virus, Nipah virus, SARS, MARS and recently the deadly Novel Coronavirus (COVID-19) have taken lives all over the world. In the time of emergency, human beings have suffered for years to prevent such outbreaks. Things would not have gotten worse if there was some tool to predict when such outbreaks will hit the most. The aim of this model is to give an estimation about when the coronavirus cases will be zero per Country and State. This model takes Hubei’s trends and applies them in each state. Because Hubei’s is the only state that has been successfully reached zero cases for a really long time. For our model, we assume that other countries and states will follow Hubei’s trend. We assume that current conditions will remain the same. We do not take into account the following: Second wave, weather, population, social distancing nature, vaccine discovery. With that aim in mind, we are proposing a model that analyzes the previous outbreak data and predicts the trend outbreaks of Covid-19 with precision.

REFERENCES APPENDIX A

References

[1] S. V. S. Vinitha, "Disease Prediction Using Machine Learning Over Big Data," in *Computer Science & Engineering: An International Journal (CSEIJ)*, February 2018.

[2] P. D. S. Kaviani, "Short Survey on Naive Bayes Algorithm," in *International Journal of Advance Research in Computer Science and Management*, 2017/11/01.

[3] C. J. K. Siriyasatien, "Dengue Epidemics Prediction: A Survey of the State-of-the-Art Based on Data Science Processes," in *IEEE Access*, 19 September 2018.

[4] L. X. Liyue Fan, "Real time aggreagate monitoring with differential privacy," in *21st ACM nternational conference on information and knowledge management*, October 2012.

[5] C. Yuanyuan, "Linear regression analysis of COVID-19 outbreak and control in Henan province caused by the output population from Wuhan," 8th May 2020.

[6] R. C. J. &. H. A. Sujath, "A machine learning forecasting model for COVID-19 pandemic in India.," *Stochastic Environmental Research and Risk Assessment,* 30th May 2020.

[7] Pandey, Gaurav & Chaudhary, Poonam & Gupta, Rajan & Pal, Saibal,"SEIR and Regression Model based COVID-19 outbreak predictions in India.," *Covid-19 Research,* April 2020.

[8] Lee, Eunji & Choi, Chang & Lee, Minchang & Oh, Kunseok & Kim, Pankoo, "An Approach for Predicting Disease Outbreaks Using Fuzzy Inference among Physiological Variables," in *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Fukuoka, Japan, 6-8 July 2016.